|  |  |  |
| --- | --- | --- |
| **Selection Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 499500 | 0.03196430206298828 |
| **2,000 (observed)** | 1999000 | 0.12699556350708008 |
| **4,000 (observed)** | 7998000 | 0.500007152557373 |
| **8,000 (observed)** | 31996000 | 1.9831328392028809 |
| **16,000 (observed)** | 127992000 | 7.931485176086426 |
| **32,000 (observed)** | 511984000 | 31.654771327972412 |
| **100,000 (estimated)** | 4999950000 | 320.7 |
| **500,000 (estimated)** | 1.25e^11 | 8007.99 |
| **1,000,000 (estimated)** | 5e^11 | 32032 |
| **10,000,000 (estimated)** | 5e^13 | 3203202.883 |

|  |  |  |
| --- | --- | --- |
| **Insertion Sort** | | |
| **List Size** | **Comparisons** | **Time (seconds)** |
| **1,000 (observed)** | 246992 | 0.028999805450439453 |
| **2,000 (observed)** | 1016724 | 0.11999368667602539 |
| **4,000 (observed)** | 3991272 | 0.48399782180786133 |
| **8,000 (observed)** | 16104203 | 1.9433999061584473 |
| **16,000 (observed)** | 64651458 | 7.700195789337158 |
| **32,000 (observed)** | 257475128 | 30.763516187667847 |
| **100,000 (estimated)** | 2499975000 | 293.5 |
| **500,000 (estimated)** | 6.4e^11 | 7338 |
| **1,000,000 (estimated)** | 2.5e^11 | 29353 |
| **10,000,000 (estimated)** | 2.5e^13 | 2935317.45 |

1. Which sort do you think is better? Why?

The insertion sort is better in number of comparisons and time

1. Which sort is better when sorting a list that is already sorted (or mostly sorted)? Why?

The insertion sort is faster in mostly sorted because its becomes closer to an O(n) algorithm while the selection sort stays at O(n^2)

1. You probably found that insertion sort had about half as many comparisons as selection sort. Why? Why are the times for insertion sort not half what they are for selection sort? (For part of the answer, think about what insertion sort must do more of compared to selection sort.)

The insertion sort only compares to a portion of the list every time and stops when it finds the right spot while the selection sort looks through each element to the right of the spot it is checking each time. While this reduces the number of comparisons the insertion sort must move each part of the array over each time it finds an out of order entry